现代 JavaScript 教程

1、New操作符具体干了什么?

a) 创建实例对象，this变量引用该对象，同时还继承了构造函数的原型；

b) 属性和方法被加入到this引用的对象中；

c) 新创建的对象由this所引用，并且最后隐式的返回 this。

2、下面程序输出结果是?

```
for (i=0, j=0; j<6, i<10; i++, j++) {
    k = i + j;//9  +  9
 }
 console.log(k);//18
//，运算符返回值是最后那个表达式的值；
```

3、如何检查一个变量是String类型

- typeof(str) === ‘string’
- Object.prototype.toString.call(str);//[object String]
- str.__proto__.constructor === String

4、["1", "2", "3"].map(parseInt) 为何返回[1,NaN,NaN]

解惑 ["1", "2", "3"].map(parseInt) 为何返回[1,NaN,NaN]_清枫草塘的博客-CSDN博客

parseInt() 函数可解析一个字符串，并返回一个整数。

语法

```
parseInt(string, radix)
```

参数	描述
string	必需。要被解析的字符串。
radix	可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。 如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以 “0x” 或 “0X” 开头，将以 16 为基数。 如果该参数小于 2 或者大于 36，则 parseInt() 将返回 NaN。

返回值

返回解析后的数字。

5、js:将数组扁平化并去除其中重复数据，最终得到一个升序且不重复的数组/二维数组去重

解惑 ["1", "2", "3"].map(parseInt) 为何返回[1,NaN,NaN]_清枫草塘的博客-CSDN博客

parseInt() 函数可解析一个字符串，并返回一个整数。

语法

```
parseInt(string, radix)
```

参数	描述
string	必需。要被解析的字符串。
radix	可选。表示要解析的数字的基数。该值介于 2 ~ 36 之间。 如果省略该参数或其值为 0，则数字将以 10 为基础来解析。如果它以 “0x” 或 “0X” 开头，将以 16 为基数。 如果该参数小于 2 或者大于 36，则 parseInt() 将返回 NaN。

返回值

返回解析后的数字。

https://blog.csdn.net/weixin_40688217

6、Set方法去重

- new一个Set对象，set方法去重，set方法只能去重一维数组
- flat方法扁平化，Infinity规定数组长度的限制
- sort方法正序排列，

- 最后把对象转化成数组：

```
var arry = [1,25,15,[1,2,15,5],15,25,35,1];
var set = new Set(arry.flat(Infinity).sort((a,b)=>(a-b)));
var arr= [...set];

    console.log(arr)// [1, 2, 5, 15, 25, 35]
```

7、 ['1', '2', '3'].map(parseInt) 输出什么?并解释

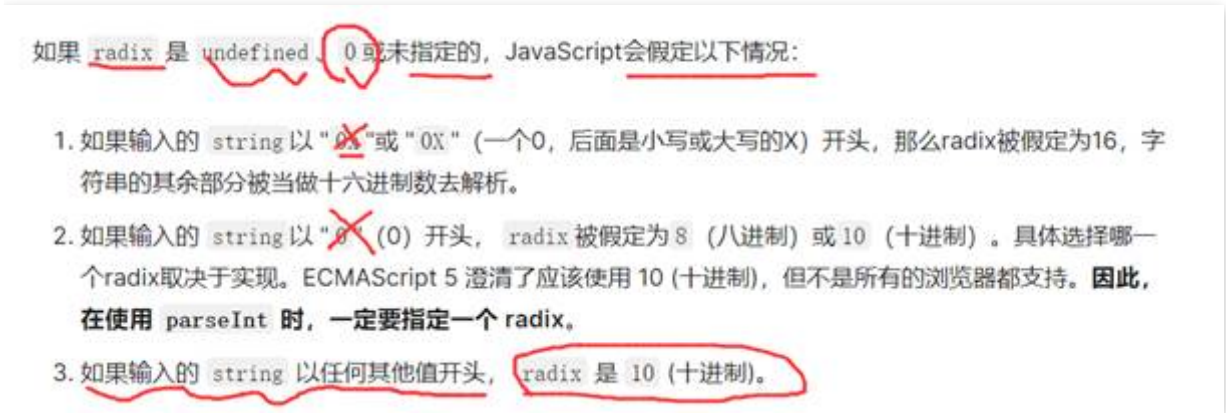
1 、map是用用来遍历的对数组的每个元素进行处理；

2、map参数是一个函数；

3、map的返回值是一个新的数组；

```
> ['1', '2', '3'].map(parseInt)
< ▶ (3) [1, NaN, NaN]
```

1, parseInt（‘1’，0)

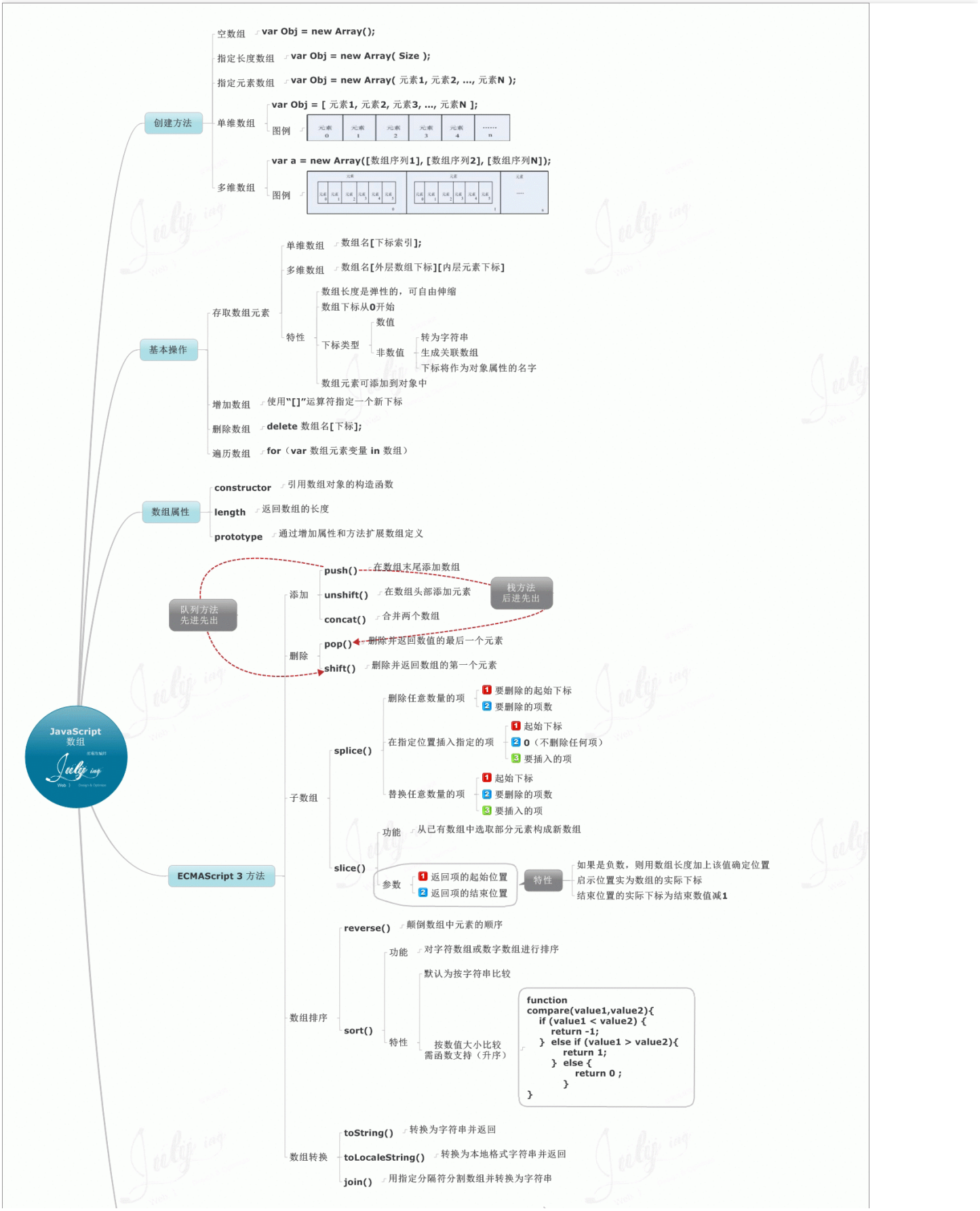


2, parseInt（‘2’，1)

3, parseInt（‘3’，2)

以下例子均返回 NaN：

```
parseInt("Hello", 8); // 根本就不是数值
parseInt("546", 2);   // 除了"0、1"外，其它数字都不是有效二进制数字
```



8、描述typeof和instanceof的区别

Typeof：获取变量类型，返回值是字符串；

Instanceof：用于检查一个对象是否是一个构造函数的实例化对象；

9、描述Array.slice和Array.splice的区别

直接看官方文档

a) slice

-- “读取”数组指定的元素，不会对原数组进行修改。

语法：

arr.slice (start, end)

start 指定选取开始位置（含），end 指定选取结束位置。

b) splice “操作”数组指定的元素，会修改原数组，返回被删除的元素。

语法：

arr. splice(index, count, [insert Elements])

index：操作的起始位置，count = 0插入元素，count > 0删除元素；

[insert Elements] 向数组新插入的元素。

面试题：

1

```
for (i=0, j=0; j<6, i<10; i++, j++) {
        k = i + j;
}
console.log(k); //18
```

- for循环的小括号里面有3个语句：初始化语句、条件判断语句、条件修改语句；
- 如果条件判断语句返回false，循环结束；

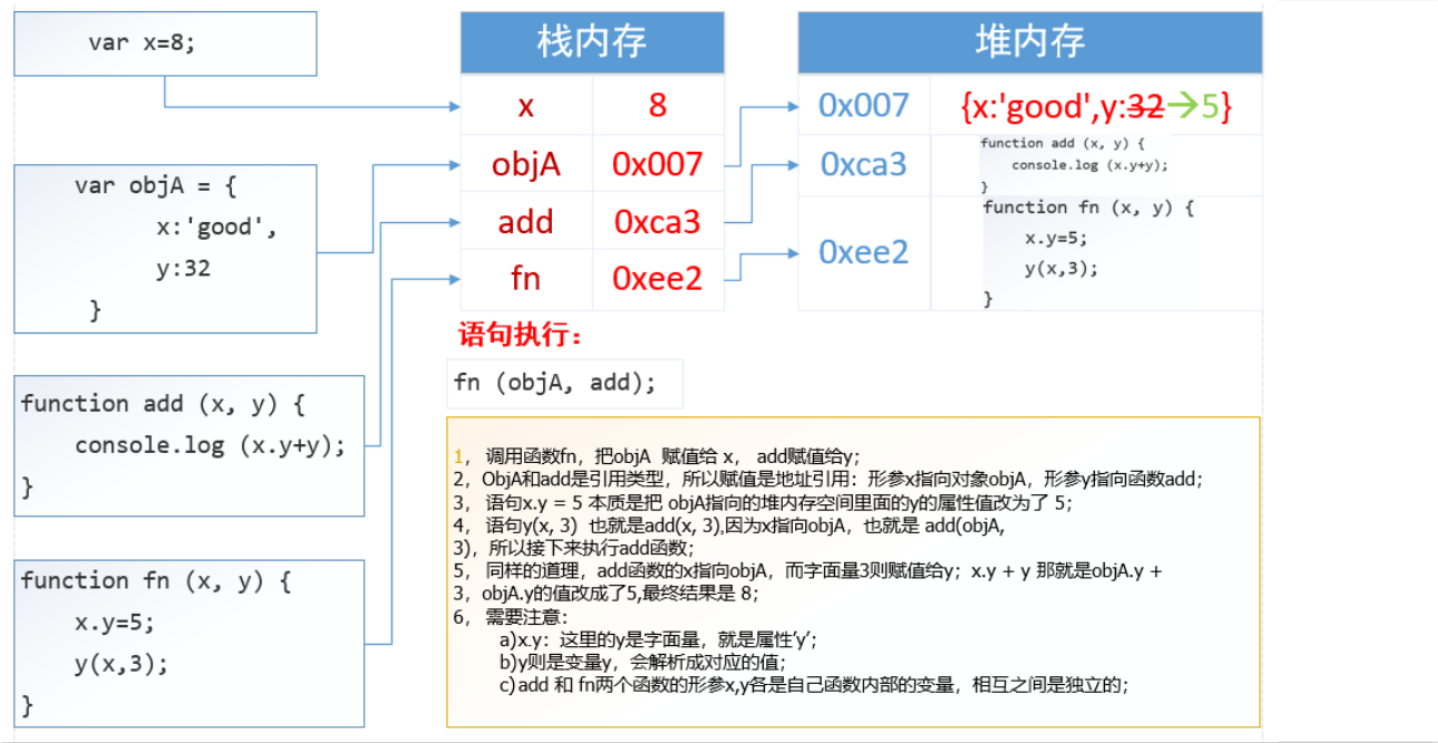
- 逗号(,)运算符返回的是最后一个表达式的值；

```
> for (i=0, j=0; j<12 || i<10; i++, j++) {
    k = i + j;
}
console.log(k);
22
```

```
> for (i=0, j=0; j<6&&i<10; i++, j++) {
    k = i + j;
}
console.log(k);
10
```

2

```
var x=8;
var objA = {
    x:'good',
    y:32
   }
function add (x, y) {
     console.log (x.y+y);
  }
function fn (x, y) {
     x.y=5;
     y(x,3);
  }
   fn (objA, add);
```



3

```
var buttons = [{name:'b1'},{name:'b2'},{name:'b3'}];
            function bind (){
                for (var i = 0; i < buttons. length; i++) {
                   buttons[i]. onclick = function () {
                       console.log(i);
                   }
                }
            };
            bind ();
           console.log(buttons);
            buttons [0]. onclick ();//3
            buttons [1]. onclick ();//3
            buttons [2]. onclick ();//3
```

- var声明的变量没有块级作用域;
- 函数内部循环的过程 只是 给每个对象追加一个onclick属性并赋值为函数而已，一开始是没有执行的;
- 函数名后面跟上小括号表示函数的执行

4

```
(function () {
                var a = b = 3;
```

```
})();
 console.log ("a defined? " + (typeof a !== 'undefined'));
 //a defined? false
 console.log ("b defined? " + (typeof b !== 'undefined'));
 //b defined? true
 console.log(b);
 //3
 console.log (typeof a);
 //undefined
```

- 函数内部声明的变量是局部变量；
- 如果函数内部直接给一个没有声明的变量赋值，那么该变量是全局变量；
- !== 表示不完全相等：'number' !== 'undefined' 返回true